

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Authentication and Authorization
Across Autonomous Network Systems**

Inventor(s):

Don Schmidt

Cliff Van Dyke

Paul Leach

Praerit Garg

Murli Satagopan

ATTORNEY'S DOCKET NO. MS1-680US

TECHNICAL FIELD

This invention relates to networked systems and, in particular, to authentication and authorization across autonomous network system boundaries.

BACKGROUND

A corporation having several business entities may want to administratively separate individual user and computer accounts associated with the separate businesses for such reasons as security, administrative control, budget separation, geographic affinity, political separation, and the like. Each business entity can be implemented as a domain which is a networked group of interconnected computing entities such as servers and clients. Each domain can be structured as a secured unit typically having a computer implemented as a domain controller to locally administrate network access and domain functions. A corporation is only one example of any company, organization, or enterprise having separable divisions and sub-divisions that are setup as independent and secure domains.

To manage closely related business entities, such as within a corporation, each respective domain can be interconnected within a "forest". When domains are grouped together and implemented as a network system in a forest, the forest boundary becomes the trust (e.g., security) boundary and the unit of administrative isolation for the group of domains. Such a configuration can be implemented with Active Directory™ which is an enterprise-wide directory service in Windows® 2000. Windows® 2000 is an operating system available from Microsoft Corporation of Redmond, Washington.

For the reasons that an enterprise having separable divisions may implement each division as a separate domain, the enterprise may also want to

1 implement multiple forests. In particular, administrative autonomy and asset
2 isolation are reasons to implement multiple forests. Administrative autonomy may
3 be desired if separate divisions do not trust another's administrators for political or
4 security reasons, or if the divisions cannot agree on common change control or
5 configuration policies. Asset isolation may be desired if separate governmental
6 agencies and/or business divisions are conglomerated via mergers and
7 acquisitions, yet wish to maintain separate and independent network system
8 infrastructures for security or budgetary reasons.

9 In a distributed network-wide directory service, enterprise-wide address
10 lists, calendars, schedules, distribution lists, and the like are not supported across
11 forest boundaries without manual directory synchronization of user accounts.
12 When administrative autonomy or asset isolation is required, directory
13 synchronization may not be possible due to schema differences, or may not be
14 allowed because of the personnel information that would be disclosed.

15 The usability and manageability afforded by the cohesiveness of multiple
16 domains in a single forest is given up when implementing multiple forests to attain
17 secure isolation for enterprise divisions. In many cases, however, resources need
18 to be shared between the different forests of a distributed enterprise. An email
19 system serving all divisions of a particular enterprise is an example of an
20 application that requires multiple forest access authorization. Additionally, users
21 that travel between geographically separated divisions within an enterprise need to
22 be able to logon with their credentials at a remote forest domain and access
23 resources throughout an enterprise.

24 In a single forest, a trust link between two domains enables security
25 principals from one domain to be authenticated in another domain. When a first

1 domain is configured to trust a second domain, the first domain is “trusting” and
2 the second domain is “trusted”. The first domain trusts the second domain to
3 authenticate users of the second domain when the users attempt to gain access to
4 protected resources in the trusting, or first, domain. The trusted domain makes its
5 accounts available to be used in the trusting domain. The trusting domain allows
6 the trusted domain to return a list of account and group membership information to
7 identify users authenticated by the trusted domain.

8 Multiple forests do not inherently trust each other. With conventional
9 networked systems, it is difficult to manage a trust link across multiple forests
10 because there is no provision to establish trust across different forest boundaries.
11 Currently, the only type of trust supported between domains in separate forests is
12 “external trust”. This is direct, non-transitive trust between two domains in
13 separate forests. An administrator has to manually establish a separate trust link
14 between every pair of domains in two forests if a user from any domain in the first
15 forest is going to logon to a computer from any domain in the second forest.
16 However, establishing a mesh of direct trust links between all of the domains of
17 multiple forests is an unmanageable and onerous task for system administrators.

18 Except for manually established direct domain-to-domain trust links, it is
19 not possible to perform such tasks as accessing shared resources across multiple
20 forest boundaries. Without being able to establish a trust link between multiple
21 forests, it is not known where to route authentication and/or authorization requests
22 that can be serviced by domains in other forests.

23 Authentication is the process of verifying the identity of a security principal
24 when access to a secured resource is requested. The verification process can be
25 applied to users, computers, and/or services executing in the security context of a

1 user or computer. Typically, user authentication is implemented in either of two
2 ways. One way is to associate a username with a password and require both the
3 username and password at the time of an initial request to access a network
4 system. A second way is to use secure access tokens granted by an operating
5 system to authentic users.

6 Once authentication has been accomplished, authorization is the process of
7 determining whether a security principal is allowed to perform a requested action.
8 Authorization uses information about the security principal to determine which
9 resources the security principal can access. A common technique consists of
10 comparing security identifiers that represent the security principal and associated
11 group memberships with an access control list that specifies the identities that may
12 access a given resource, and what type of access is allowed.

13 Kerberos is one example of a secure method for mutually authenticating
14 both users and services in a computer network. Kerberos allows a user to obtain
15 an encrypted ticket-granting-ticket that can then be used to request a service ticket
16 for a particular service on a network server. With a service ticket, the user's
17 password does not have to pass through the network. A Kerberos ticket provides a
18 secure way to transport an encryption key that is shared between a user and a
19 server for authentication across a potentially non-trusting network.

20 To get a ticket-granting-ticket, Kerberos authenticates a user from an
21 authentication server. The authentication server creates the encryption key to be
22 shared between the user and a ticket granting service. Two copies of this
23 encryption key are returned to the user, one of which is encrypted in the user's
24 master key (a key derived from the user's password) and the other which is placed
25 in the ticket-granting-ticket to be encrypted in the master key of the ticket granting

1 service. The ticket-granting-ticket is then sent to the ticket granting service along
2 with a request for a service ticket for a particular server or service on the network.
3 The ticket granting service returns the service ticket that can be sent to a network
4 server for the requested service or resource access. When the user attempts to
5 logon to the server, the service ticket is provided with an authenticator (a Kerberos
6 data structure encrypted under the same session key that was placed in the service
7 ticket). When the server receives a service ticket and an authenticator, the server
8 has enough information to authenticate the user. A subsequent network exchange
9 can be performed to enable the user to authenticate the server.

10 A ticket-granting-ticket is time-stamped to allow a user to make additional
11 requests using the same ticket within a certain time period without having to be
12 reauthenticated. Issuing a valid ticket for a limited time period makes it less likely
13 that a second user will later be able to acquire and use the ticket inappropriately.

14 Fig. 1 shows a conventional network architecture 100 representing an
15 enterprise having two separable divisions implemented as forests *A* and *B*. Each
16 forest *A* and *B* is an administratively isolated network system 102 and 104,
17 respectively. Network system 102 has two domains 106(1) and 106(2) each
18 having a computer implemented as a domain controller 108(1) and 108(2),
19 respectively. The two domains 106 form a domain tree with a bi-directional trust
20 link 110 that is automatically established when an administrator creates a second
21 domain, such as domain 106(2). A domain tree is established with multiple
22 domains and forms a contiguous namespace.

23 The domain controllers 108 in forest *A* implement a network-wide
24 partitioned directory, such as Active Directory™ which is an enterprise-wide
25 directory service in Windows® 2000. Windows® 2000 is an operating system

1 available from Microsoft Corporation of Redmond, Washington. The domain
2 controllers 108 can also implement other directory services, such as NDS
3 eDirectory available from Novell, an iPlanet directory service available from Sun
4 Microsystems Inc., or the like. Each domain controller 108 in a separate domain
5 106 of the network system 102 maintains a copy of a partition of the directory
6 which typically contains those objects that are pertinent only to a particular
7 domain. Pertinent objects include those that facilitate the administration of
8 security principals' authentication, authorization, and network access at a
9 particular domain controller.

10 Domain 106(1) includes a global catalog server 112 that maintains
11 network-wide information for network system 102 and is communicatively linked
12 to the domain controllers 108 via a network communications system (not shown).
13 In a network configuration, a global catalog server can be implemented to
14 maintain a directory of all the user and group memberships within the network for
15 each user and group account authorized to access the network. Global catalog
16 server 112 provides a central information source that can be accessed by domain
17 controllers 108 to locate and access network-wide resources upon user request.
18 Network system 102 has a workstation 114 connected to domain controller 108(1)
19 to facilitate a user request to access network system 102.

20 Similar to network system 102, network system 104 is an administratively
21 isolated forest. Network system 104 has two domains 116(1) and 116(2) each
22 having a computer implemented as a domain controller 118(1) and 118(2),
23 respectively. The two domains 116 form a domain tree with a bi-directional trust
24 link 120. The trust link 120 between the two domains 116 enables a user with an
25 account in one domain to have access to resources in another domain within the

1 boundary of forest *B*. When trust links are established between domains, user and
2 group objects from the directory can be given access rights and permissions in
3 domains other than the domain where these objects are located.

4 Domain 116(1) includes a global catalog server 122 that maintains
5 network-wide information for network system 104 and is communicatively linked
6 to the domain controllers 118 via a network communications system (not shown).
7 Network system 104 also has a resource server 124 that maintains network-wide
8 accessible resources. The resources are only available within forest *B*, however,
9 because of the administrative isolation from forest *A*.

10 It is possible for administrators to manually create an explicit trust link
11 between domains in separate forests. However, even when creating a sufficient
12 trust link, Kerberos authentication between forests frequently fails. The primary
13 cause is that either the username, or the service name, cannot be resolved by a
14 domain controller or global catalog server in the forest where the logon request
15 originates. This causes Kerberos authentication to fail for both interactive and
16 network logon requests when the user and service accounts are managed in
17 different forests.

18 For example, if a user at workstation 114 in forest *A* requests access to the
19 resource server 124 in forest *B*, the Kerberos service ticket request will fail. When
20 workstation 114 at forest *A* sends a Kerberos service ticket request for resource
21 server 124 to domain controller 108(1), the domain controller will not find the
22 service name in its local database. It then queries the global catalog server 112 in
23 forest *A* for the resource server 124. The global catalog server 112, however, does
24 not recognize the requested service name either. Thus, Kerberos authentication
25 fails.

1 If both the workstation 114 and resource server 124 support a common
2 operating system authentication protocol, the workstation and resource server can
3 negotiate authentication via an external trust relationship so that a logon request
4 can succeed. However, conventional operating system authentication protocols do
5 not provide equivalent Kerberos functionality, such as mutual authentication,
6 and/or delegation. Therefore, a user can not access a resource in a forest that is
7 beyond the security boundary of the user's home forest if the connection requires
8 mutual authentication or delegation.

9 10 SUMMARY

11 A network system indicates which network domains it claims to manage
12 within its secured boundary. An enterprise network system can establish a trust
13 link between two autonomous network systems that enables security associations
14 and transitive resource access between network domains of the two network
15 systems. The trust link enables each network system to maintain a secured
16 boundary, yet share resources and authenticate network access requests across the
17 network systems boundaries.

18 The trust link is defined by data structures maintained by a domain
19 controller in each of the respective network systems. When the trust link is
20 initially defined, a data structure is created on a single domain controller in each
21 network system. Each domain controller can then replicate its data structure
22 within the domain controller's respective network system. A trust link can be
23 established as a one-way trust relationship or as a two-way trust relationship. For
24 a one-way trust link, a domain controller in a trusting first network system stores
25 network system identifiers corresponding to a trusted second network system.

1 Similarly, a domain controller in a trusted domain in the second network system
2 stores network system identifiers corresponding to the trusting first network
3 system. A two-way trust link is established as a pair of one-way trust links, and
4 the data structures maintained by each of the network systems identify both the
5 outgoing and incoming trust relationships for each network system.

6 A user having an account maintained in the second (i.e., trusted) network
7 system can interactively logon to a server having an account that is maintained in
8 the first (i.e., trusting) network system. A domain controller in the first network
9 system determines from an established trust link with the second network system
10 where to communicate an authentication request received from an account
11 managed in the second network system. The first network system can authorize
12 the account access to the resource even though the account is managed and
13 authenticated by the second network system.

14 Additionally, a network logon request to a server managed in the first (i.e.,
15 trusting) network system can be initiated from the second (i.e., trusted) network
16 system. Implementing Kerberos authentication, the second network system
17 attempts to determine from the trust link where to communicate the service ticket
18 request. If the request is successful, the client in the second network system can
19 send the ticket to a server in the first network system to complete single, or mutual
20 authentication.

BRIEF DESCRIPTION OF THE DRAWINGS

The same numbers are used throughout the drawings to reference like features and components.

Fig. 1 is a diagram of a conventional network architecture having separable divisions implemented as two independent and separate networks.

Fig. 2 is a diagram that illustrates an enterprise multi-forest network architecture having established cross forest trust links.

Fig. 3 illustrates a data structure of records that include forest trust information to define a forest trust link.

Fig. 4 is a flow diagram of a method for establishing a trust link between autonomous network systems.

Fig. 5 is a flow diagram of a method to authenticate a network logon request across autonomous network system boundaries.

Fig. 6 is a flow diagram of a method to filter domain security identifiers across autonomous network system boundaries.

Fig. 7 is a diagram of computing systems, devices, and components in an environment that can be used to implement the invention described herein.

DETAILED DESCRIPTION

Introduction

The following describes systems and methods to manage and control namespaces across independent and secured network systems without requiring an all-encompassing directory service to join the separate network systems together. This enables an enterprise to establish independent and administratively secure

1 network systems for separable divisions of the enterprise, yet provide access to
2 shared resources across the independent network systems.

3 Exemplary Network Architecture

4 Fig. 2 illustrates a network architecture 200 representing an enterprise
5 having three separable divisions implemented as forests *A*, *B*, and *C*. Each forest
6 *A*, *B*, and *C* is an administratively isolated and independent network system 202,
7 204, and 206, respectively. Although network architecture 200 is illustrated
8 having only three forests (i.e., independent networks), the systems and methods
9 described herein are applicable to a network architecture having any number of
10 autonomous networks of varying configuration. See the description of
11 “Exemplary Computing System and Environment” below for specific examples
12 and implementations of network and computing systems, computing devices, and
13 components that can be used to implement the network architectures described
14 herein.

15 Network system 202, which is forest *A*, has three domains 208(1), 208(2),
16 and 208(3) each having at least one computing device implemented as a domain
17 controller 210(1), 210(2), and 210(3), respectively. The three domains 208 form a
18 domain tree and represent a hierarchically contiguous namespace. A namespace is
19 a grouping of related names or identifiers that symbolically represent a location of
20 information, data elements, or other network accessible resources. A
21 hierarchically contiguous namespace is a namespace that is partitioned across
22 multiple domains that are hierarchically related, such as domains 208(1), 208(2),
23 and 208(3).

24 Domain 208(1) is associated with domain 208(2) by an explicit two-way
25 trust link 212, and domain 208(2) is associated with domain 208(3) by an explicit

two-way trust link 214. An example of the namespaces representing the three domains 208 is domain1.com for domain 208(1), domain2.domain1.com for domain 208(2), and domain3.domain2.domain1.com for domain 208(3), where domain 208(1) is the root of the domain tree and, in this instance, is also the root domain for forest *A*.

Furthermore, domains 208(1) and 208(3) are transitively associated by virtue of their respective explicit trust links with domain 208(2). The Windows® 2000 operating system, for example, automatically establishes bi-directional, transitive trust links such as between domains 208(1) and 208(3) in a single forest. That is, if domain 208(1) trusts domain 208(2) by way of established trust link 212, and domain 208(2) trusts domain 208(3) by way of established trust link 214, then domain 208(1) transitively trusts domain 208(3). The domains in a forest form a hierarchically contiguous namespace, as well as a transitive trust relationship, for the purpose of serving authentication and authorization requests.

In Fig. 2, forests *A*, *B*, and *C* could each have one or more domain trees. Although a domain tree represents a single, contiguous namespace, a forest itself does not necessarily represent a single, or distinct, namespace. That is, two or more domain trees in a forest do not have to form a contiguous namespace. The domain at the root of the forest identifies the name of the forest and identifies the root of the two-way, transitive trust relationships between all of the domain trees in the forest. For example, domain 208(1) is the root domain of forest *A*.

Domain 208(1) includes a global catalog server 216 that maintains network-wide information for network system 202 and is communicatively linked to the domain controllers 210 via a network communications system (not shown).

1 In a network configuration, a global catalog server can be implemented to
2 maintain a directory of all user and their group memberships within the forest for
3 each user authorized to access the network. The global catalog server 216
4 provides a central information source that can be accessed by the domain
5 controllers 210 to locate and access network-wide resources upon user or
6 application request.

7 Network system 202 also has a workstation 218 connected to domain
8 controller 210(2) to facilitate a user request to access resources in network system
9 202. Although network system 202 is illustrated having only one workstation 218,
10 the systems and methods described herein are applicable to a network system and
11 a network architecture having any number of workstations connected to any of the
12 domain controllers. In this instance, work station 218 facilitates user, client,
13 application, or account access to the resources of network architecture 200 via
14 domain controller 210(2). Although the following description pertains mainly to
15 user requests to access a network system and resources maintained throughout a
16 network architecture, it is to be appreciated that any type of account, such as a
17 user, client workstation, application, service, server, and the like can be
18 implemented within a network architecture to request access to network and
19 server-based resources.

20 Similar to network system 202, network systems 204 and 206 are
21 administratively isolated forests *B* and *C*, respectively. Network system 204 has
22 two domains 220(1) and 220(2) each having a computing device implemented as a
23 domain controller 222(1) and 222(2), respectively. The two domains 220 in forest
24 *B* form a domain tree and are associated by an explicit trust link 224.
25

1 Domain 220(1) includes a global catalog server 226 that maintains
2 network-wide information for network system 204 and is communicatively linked
3 to the domain controllers 222 via a network communications system (not shown).
4 Network system 204 also has a network resource server 228 that maintains
5 network-wide accessible resources. Network system 206 is illustrated as a single
6 domain forest *C* and has a global catalog server 230.

7 Namespaces

8 A network system manages different namespaces that identify different
9 types of network domain components such as users, computers, applications,
10 COM objects, and the like, within a network architecture. A namespace is
11 identified by the range of names that it contains, some of which are used to
12 communicate authentication and/or authorization requests to a trusted domain
13 when a security principal name cannot be resolved locally. Examples of such
14 namespaces include namespaces constructed for a domain tree name, a user
15 principal name, a service principal name, or for specific identifiers associated with
16 a specific domain, such as a domain's domain name system (DNS), Netbios name,
17 or its Security Identifier (SID).

18 A domain tree name identifies a domain tree in a forest as a hierarchical
19 contiguous namespace. Domain tree names in a forest are derived from the name
20 of the root domain of the forest. In Fig. 2, an example of a domain tree name is
21 domain1.com, where domain 208(1) is also the root domain of forest *A*.

22 A user principal name (UPN) identifies an entity to a security system and
23 can identify a user logged onto a network, or an application or process executing
24 on a computing device. A user principal name is one type of a name that identifies
25 a particular user. A UPN is composed of a prefix, which is the user's logon name,

1 followed by the “@” symbol, and a suffix that identifies the namespace to which
2 the UPN belongs, such as the department in which the user works, or the domain
3 where the user account is maintained. For example, someone in the company.com
4 domain can have the username, someone@company.com. The UPN suffix is the
5 component of the username to the right of the rightmost “@” symbol, which in
6 this example is “company.com”.

7 A service principal name (SPN) identifies a particular instance of a service
8 running on a specific computer. An SPN typically consists of a prefix that
9 identifies the service, and a suffix that identifies a computer on which the service
10 instance is executing. The SPN suffix can consist of a name that identifies the
11 host computer, or it can also include a component that identifies the domain name
12 to which the computer is connected.

13 A domain identifier (domainID) is a three part name that identifies a
14 particular domain. The three component parts of a domainID are the domain’s
15 DNS name, Netbios name, and domain security identifier (domain SID). A
16 security identifier is a fixed numerical that uniquely identifies a domain, or
17 security principal (e.g., user, group, or a service) that is a member of the domain.
18 A security identifier also includes a component that identifies the authority issuing
19 the security identifier, such as an operating system.

20 **Exemplary Trust Links Between Autonomous Network Systems**

21 Fig. 2 illustrates forest trust links between forests *A* and *B*, and between
22 forests *A* and *C*. Forest trust links are established between the root domains of two
23 network systems, such as a two-way trust link 232 between forest *A* and forest *B*.
24 A trust link between two network systems enables transitive security associations
25 and resource access between the domains of the two network systems. When trust

1 link 232 is established, all of the domains 208 in forest *A* automatically trust all of
2 the domains 220 in forest *B*, and vice-versa. With the two-way trust link 232,
3 accounts (e.g., users) in either forest *A* or *B* can be authenticated and access
4 resources in the other forest as if they were a user in that forest.

5 A one-way trust link 234 is established between forest *A* and forest *C*. In
6 this example, forest *C* is the trusting forest and forest *A* is the trusted forest. With
7 the one-way trust link 234, users having accounts in forest *A*, the trusted forest,
8 can access resources in forest *C*, the trusting forest. That is, users having accounts
9 in forest *A* can be granted permissions and access rights in forest *C* without an
10 account in forest *C*.

11 To establish a forest trust link, an administrator for each respective network
12 system initiates a trust link with another network system. For example, to
13 establish the two-way trust link 232 between forest *A* and forest *B*, an
14 administrator for forest *A* initiates the trust link with forest *B*, and an administrator
15 for forest *B* initiates the trust link with forest *A*. However, all of the security
16 associations between the domains of the respective network systems are
17 automatically established by a computing device in the root domain of each
18 network system.

19 Forest trust links include constraints that enable network system
20 administrators to control the trust afforded to individual namespaces managed by
21 the trusted network system domains. A forest “publishes”, or identifies, all of the
22 namespaces that it manages. An administrator in a trusting forest can configure
23 which namespaces the trusting forest actually trusts a trusted forest to be
24 authoritative for – that is, which names the trusting forest trusts another forest to
25 authenticate.

1 When a forest trust link is created, the trusting forest obtains the
2 namespaces that the trusted forest publishes and claims to manage. Whether the
3 trust link is one-way or two-way, a domain controller in the root domain of each
4 forest creates a trusted domain object that defines the forest trust link between the
5 local forest and the remote forest. Forest trust information is stored in a trusted
6 domain object to identify the namespaces that a remote forest publishes and claims
7 to manage. Each record in a trusted domain object includes a field to indicate
8 whether the local forest accepts or rejects a remote forests particular namespace.
9 If a namespace is accepted, the local forest trusts the remote forest to be
10 authoritative for the particular namespace.

11 **Exemplary Trusted Domain Object**

12 Fig. 3 illustrates an exemplary data structure 300 that can be implemented
13 as part of a trusted domain object to define a forest trust link between two
14 autonomous network systems. Exemplary data structure 300 illustrates how
15 namespaces can be maintained as forest trust information (FTinfo) in any number
16 of records 302 within a single trusted domain object data structure. An FTinfo
17 record 302 in data structure 300 has several fields including a namespace field
18 304, a namespace data field 306, a flags field 308, a timestamp field 310, and a
19 pointer field 312.

20 Each of the record fields 304 through 312 can contain any numerical or
21 alphanumerical value that uniquely identifies the data in the fields. Additionally,
22 the combination of records and fields shown in data structure 300 is merely an
23 example to illustrate maintaining forest trust information. Those skilled in the art
24 will recognize that any combination of records, fields, and data can be created and
25 defined in a data structure.

1 An FTinfo record 302 stores one of three types of namespaces in a
2 corresponding namespace field 304. Two of the namespace types, a top level
3 name and a domain identifier (domainID), represent namespaces that a forest
4 explicitly claims to manage. The third namespace type, an "exclusion", is an
5 artificial construct utilized to segment a hierarchical namespace so that a subtree
6 can be managed by a different forest than the one which manages the top level of
7 the namespace.

8 A namespace field 304 identifies the type of namespace that a trusted forest
9 publishes, or in the case of an exclusion, identifies a restriction on a namespace
10 that the trusted forest publishes. A namespace data field 306 describes the value
11 of a corresponding namespace 304. A flags field 308 indicates if a particular
12 namespace 304 is trusted or not, based either on a collision with a trusted
13 namespace in another trusted domain object, or on explicit rejection by an
14 administrator. If a local forest indicates that a particular namespace 304 is to be
15 trusted, the local forest considers the remote forest to be authoritative for the
16 namespace. A timestamp field 310 indicates when a corresponding namespace 304
17 is trusted in a trusted domain object. A pointer field 312 stores a pointer to the
18 forest that includes the corresponding namespace 304.

19 A first type of namespace 304 is a top level name which is a network
20 system identifier that describes a hierarchical namespace that is published by a
21 trusted forest. A top level name includes all subordinate domain subtree names,
22 unless a subtree is explicitly excluded by an exclusion record in the same forest
23 trusted domain object. Domain tree names, service principal name suffixes, and
24 user principal name suffixes are all stored as a top level name. The namespace
25 data 306 corresponding to a top level name of namespace type 304 is the string

1 name that identifies the namespace. An example of a top level name is record one
2 (1) in data structure 300.

3 A second namespace type 304 is a domain identifier (domainID) that
4 identifies an existing domain in a trusted forest. A domainID is subordinate to a
5 top level name in the same forest trusted domain object. If a top level name is
6 flagged, or otherwise identified, as not trusted, all of the domainID records
7 identifying a subordinate domain in the same forest will automatically not be
8 trusted, irrespective of their individual flag settings.

9 The namespace data 306 corresponding to a domainID namespace type 304
10 is the three component parts of a domainID which are the DNS name, Netbios
11 name, and domain SID. The Netbios name has a corresponding trust flag that can
12 be set independently of a trust flag 308 for a particular domainID record. Domain
13 SIDs are used to filter authorization data that is returned from a forest via a trust
14 link. Netbios domain names are used to determine where to route, or otherwise
15 communicate, authentication and authorization requests when complete DNS
16 names are not available. An example of a domainID is record three (3) in data
17 structure 300.

18 A third namespace type 304 is a top level name exclusion record. An
19 exclusion record excludes a subtree from the trusted namespace associated with a
20 hierarchical top level name record so that the namespace defined by the subtree
21 can be trusted as a top level name by another forest in a trusted domain object
22 record for a different forest. The namespace data 306 corresponding to an
23 exclusion namespace type 304 is the string name for the root of the subtree and
24 includes the top node of the subtree. Exclusion records are not published by a
25

1 trusted forest, but rather are created by an administrator of the trusting forest. An
2 example of an exclusion namespace type is record four (4) in data structure 300.

3 The flags field 308 indicates that a record 302 is trusted, or enabled, if
4 "flags = 0" for a particular record. If a record 302 is enabled, the local forest
5 accepts the remote forest's claim to be authoritative for the corresponding
6 namespace 304. A record is disabled if the corresponding flag field 308 indicates
7 that a conflict exists, or that it has been disabled by an administrator or is pending
8 administrative review. The corresponding namespaces for records one through
9 three in data structure 300 are indicated as being trusted and enabled with flag
10 settings of "0". Exclusion records (e.g., record four (4)) are not disabled, but
11 rather deleted from data structure 300 by an administrator if the exclusion record
12 is no longer valid.

13 A top level name or domainID record is disabled when a namespace
14 claimed in a newly created record duplicates a trusted namespace identified by an
15 existing, enabled FTinfo record in a different forest. A conflict resolution policy
16 enables the first FTinfo record, and any subsequent records that duplicate the
17 enabled record are automatically disabled (i.e., the flags field 308 for a
18 corresponding record 302 indicates that the record is not trusted, or is disabled).

19 A top level name or domainID record can also be disabled by an
20 administrator via a trust management user interface. When a forest trusted domain
21 object is initially created, all of the FTinfo records are enabled as long as they do
22 not duplicate already existing records, as described above. When the FTinfo for a
23 particular trust link is updated, and new top level name records are generated, the
24 new records are identified as disabled, yet new. Upon review, an administrator can
25 enable the new, disabled top level name records. When new domainID records are

generated, the new records are enabled unless they are subordinate to a disabled top level name record, or a top level name exclusion record in the same trusted domain object.

Method for Namespace Collision Detection to Establish a Trust Link

Namespace collision detection is implemented to ensure that only one forest in a network architecture is trusted to be authoritative for a particular namespace. When a trust link is initiated, the trusting forest obtains the namespaces that the trusted forest publishes and claims to manage. The namespaces received from the trusted forest are not automatically trusted, but rather a collision detection process is implemented to prevent an overlap with a namespace that the trusting forest manages for itself, or already trusts another forest to manage. In addition, a network system administrator can selectively trust or not trust individual namespaces that have passed a collision detection test.

Fig. 4 illustrates a method to establish a trust link between autonomous network systems and to detect an overlap in namespaces. The described method references components of network architecture 200 (Fig. 2) and data structure 300 (Fig. 3). The order in which the method is described is not intended to be construed as a limitation. Furthermore, the method can be implemented in any suitable hardware, software, firmware, or combination thereof.

At block 400, a forest trust link is initiated from a trusting forest to a trusted forest. The forest trust link can be initiated by a network system administrator from forest 202 (trusting forest *A*) to forest 204 (trusted forest *B*). At block 402, a trusted domain object is created to define the forest trust link established between the two forests. For example, domain controller 210(1) in the root domain of the

1 trusting forest 202 creates a data structure 300 as a trusted domain object that
2 defines the forest trust link between forest *A* and forest *B*.

3 At block 404, the trusting forest receives the namespaces which are
4 network system identifiers that the trusted forest publishes and claims to manage.
5 For example, domain controller 210(1) in trusting forest 202 receives the
6 namespaces from trusted forest 204. Domain controller 210(1) maintains a cache
7 of all existing forest trust object records that local forest 202 maintains. The
8 namespaces received from forest 204 are compared to the cache of records having
9 a top level name or domainID namespace type to determine if there will be a
10 namespace collision, or overlap. Existing FTinfo records 302 that are marked as
11 not trusted (i.e., the flags field 308 indicates a conflict identifier) are ignored by
12 the collision detection process. Exclusion records are used to resolve collisions,
13 but are not checked themselves for collisions. For FTinfo records that are a
14 domainID namespace type, the three component parts are evaluated separately.

15 At block 406, domain controller 210(1) in trusting forest 202 determines a
16 namespace type 304 of a namespace received from trusted forest 204. Top level
17 name and domainID namespace types records are compared with existing FTinfo
18 records from all other trusted forests. If the namespace is determined to be a top
19 level name, the namespace is compared to any cached trusted domain object
20 records having a top level name namespace type 304 at block 408. If the
21 namespace received from forest 204 is determined to be a domainID, the
22 namespace is compared to any cached trusted domain object records having a
23 domainID namespace type 304 at block 408. A substring match of a received
24 namespace with a cached namespace is determined to be a collision if the existing
25

1 record is flagged as trusted. A substring match indicates that two hierarchical
2 namespaces partially overlap in a superior/subordinate relationship.

3 For example, if an existing record has a top level name value of
4 company.com, a received top level name value of store.company.com will be
5 determined to cause a collision with the existing record. This will not cause a
6 collision if the forest trusted domain object that contains the top level name record
7 for the superior name also contains an exclusion record for the subordinate
8 namespace.

9 The namespace data field 306 for a domainID type FTInfo record 302
10 received from a trusted forest is compared with all records of the same type from
11 all trusted forests, as well as all of the domains managed by the local, trusting
12 forest. The Netbios name and the domain SID components of a domainID are
13 compared individually. If a Netbios name and/or a domain SID collide with an
14 existing record, the flag value 308 for the corresponding new record is set to
15 indicate a conflict identifier, such as Netbios_disabled_conflict, or SID_
16 disabled_conflict, respectively. If the domain SID is determined to collide with an
17 existing record, the entire corresponding trusted domain object record 302 is not
18 trusted. If only the Netbios name collides with an existing record, the Netbios
19 name component is not trusted, but the other components of the corresponding
20 trusted domain object record 302 are trusted. The DNS name component does not
21 need to be checked for DNS name collisions because a DNS name is subordinate
22 to a top level name which has already been tested for collisions.

23 If a collision of a network system identifier is detected (i.e., "yes" from
24 block 408), the flag field 308 corresponding to the namespace type 304 in the new
25 trusted domain object record is set to indicate that the network system identifier is

1 not trusted at block 410. If a collision of a network system identifier is not
2 detected (i.e., "no" from block 408), the flag field 308 corresponding to the
3 namespace type 304 in the new trusted domain object record is set to indicate that
4 the namespace is trusted at block 412. Storing a namespace in a trusted domain
5 object and indicating that the namespace is trusted is based upon a first-come,
6 first-served model. The first instance of a namespace is trusted, and subsequent
7 instances of the namespace are not. The process of determining a namespace type
8 of a namespace received from a trusted forest (block 406), and either determining
9 to not trust or trust the namespace (blocks 410 and 412, respectively), is repeated
10 until all of the namespaces received from the trusted forest are evaluated (block
11 408).

12 **Authorization and Authentication across Network Systems**

13 When a trust link is established between autonomous network systems, the
14 trusted namespaces maintained by a trusting forest are used for routing decisions
15 to route, or otherwise communicate, authentication requests for names that cannot
16 be resolved in a local forest. The trusted namespaces are also used during
17 authorization requests when adding remote users or groups to domain local groups
18 or access control lists in the trusting forest. Creating trusted domain objects, and
19 detecting namespace collisions when establishing a trust link, prevents two
20 independent network systems from being trusted to authenticate a user from the
21 same namespace. For network logon, authentication requests are communicated
22 from a trusted forest via a trust link to a trusting forest. Authorization requests to
23 lookup users or groups for setting group memberships or access control lists
24 (ACLs) are communicated from the trusting forest to the trusted forest.
25

1 Kerberos mutual authentication and delegation are supported across the
2 autonomous network system boundaries. Kerberos is only one example of a
3 secure method for mutually authenticating both users and services in a computer
4 network. Other authentication protocols and methods, such as Digest Access
5 Authentication, Basic Authentication, SSL Authentication, and the like can be
6 used in the context of the systems and methods pertaining to trust links established
7 between autonomous network systems as described herein.

8 Lacking a unified directory service (e.g., meta-directory) neither a domain
9 controller, nor a global catalog server, in one network system can resolve a user or
10 service name from another independent network system. However, establishing a
11 trust link across autonomous network systems allows authentication and/or
12 authorization requests to succeed across the network system boundaries because a
13 network system global catalog server in the forest where the request originates can
14 generate a routing hint that allows the request to be referred to the forest that
15 manages the name, and can thus satisfy the authentication or authorization request.

16 **Method for Authentication and Namespace Resolution**

17 Fig. 5 illustrates a method to authenticate a network logon request across
18 autonomous network system boundaries. Authentication is the process of
19 verifying the identity of a security principal by submitting credentials to a domain
20 controller for validation. The described method references components of
21 network architecture 200 (Fig. 2) and data structure 300 (Fig. 3). The order in
22 which the methods are described is not intended to be construed as a limitation.
23 Furthermore, the methods can be implemented in any suitable hardware, software,
24 firmware, or combination thereof.

1 At block 500, a network system receives an authentication request to logon
2 to a second, independent network system. For example, domain controller 2,
3 which is identified as 210(2) in forest 202, receives an authentication request from
4 a user via workstation 218 to logon to domain controller 5, which is identified as
5 222(2) and managed in forest 204.

6 At block 502, domain controller 210(2) determines whether the
7 authentication request can be resolved within the local network system, forest 202.
8 If the request can be resolved within the local network system (i.e., “yes” from
9 block 502), the user is authenticated to logon to the local network system, at block
10 504.

11 If the authentication request cannot be resolved within the local network
12 system (i.e., “no” from block 502), a component of the request is compared with
13 trusted domain object FTInfo records 302 at the local network system to determine
14 if a remote network system can resolve the request at block 506. If the requested
15 name does not match a trusted namespace from any other trusted forest (i.e., “no”
16 from block 506), the user logon request is denied at block 508. If the request
17 component does match a trusted namespace in the local trusted domain object
18 FTInfo record (i.e., “yes” from block 506), the authentication request is routed, or
19 otherwise communicated, to the root domain of the trusted network system that
20 manages the trusted namespace at block 510. The forest pointer 312
21 corresponding to the matching trusted namespace FTInfo record identifies the
22 trusted network system where the authentication request is routed.

23 When the name of a security principal cannot be resolved in a local domain
24 (e.g., “no” from block 502), the system attempts to resolve the security principal’s
25 name to the forest that manages the security principal’s account. Name resolution

1 involves substring matching a component of the security principal's name against
2 external trust FTinfo records to identify a remote domain that claims to manage
3 the account for the security principal's name. A matching function isolates a
4 component of the security principal's name for comparison with FTinfo records
5 having the same namespace type. If a match is identified in a remote domain, the
6 matching function returns a routing hint of the independent network system, or
7 forest, that contains the remote domain. A match only indicates that the identified
8 remote forest claims to manage the security principal's name. The authentication
9 request is routed, or otherwise communicated, along the trust path between the
10 local and remote network systems, and if the security principal's name is actually
11 managed by the identified remote forest, the authentication request will succeed.

12 For example, a user having an account managed or maintained by domain
13 controller 5 in forest *B* can logon and be authenticated via workstation 218 in
14 forest *A*. The workstation 218 communicates a logon request to domain controller
15 2 in forest *A* and the domain controller queries global catalog server 216 in forest
16 *A* to authenticate the user. The global catalog server 216, however, does not
17 recognize the user and evaluates FTinfo records of trusted domain objects for
18 another forest to service the authentication request.

19 When a match is found, a routing hint is determined that identifies the root
20 domain of the trusted forest that manages the user's name (i.e., forest *B*).
21 Kerberos authentication requests are referred along the trust path from domain 2 in
22 forest *A* where the request originates to domain 5 in forest *B* that manages the
23 account. With Kerberos authentication, the authentication service on domain
24 controller 2 in forest *A* refers the user to an authentication service on domain
25 controller 4 in forest *B* via the trust link 232. Domain controller 4 queries global

1 catalog server 226 in forest *B* to resolve the user's name. Global catalog server
2 226 recognizes the user and domain controller 4 refers the authentication request
3 to domain controller 5.

4 Both Kerberos, and other operating system authentication protocols, use
5 routing hints to determine an independent network system, or forest, that claims to
6 manage a security principal requesting authentication. Kerberos authentication
7 requests (e.g., authentication service requests) can be referred directly to the root
8 of the trusted forest. Kerberos service ticket requests (TGS requests) are referred
9 along the trust path from an originating domain in a first forest receiving the
10 request to the root domain of the first forest. The request is then routed, or
11 otherwise communicated, via a trust link to the root domain in a second forest, and
12 then to the domain in a second forest that manages the trusted namespace.

13 Other operating system authentication protocols, such as Windows NT®
14 Lan Manager (NTLM) which is available from the Microsoft Corporation, chain
15 authentication requests for a client or user requesting authentication. For example,
16 after a user initiates an authentication request with a server, the server
17 communicates with a domain controller directly to provide user authentication
18 information to the domain controller.

19 Name resolution resolves one of four types of names: usernames, service
20 principal names, domain names, or user or group security identifiers. The name
21 type is parsed to isolate the suffix or prefix string that identifies the namespace
22 from which the name was constructed. The isolated string is then compared to
23 trusted namespaces in the similar type(s) of FTinfo records 302 maintained by
24 forests in a trusted domain object. If the isolated string does not match any trusted
25 namespace, an authentication request is denied. A denied request indicates that

1 the name cannot be resolved in the local forest, or in a trusted forest. If the
2 request does match a trusted namespace, the authentication request is routed, or
3 otherwise communicated, to the corresponding network domain.

4 User principal names are parsed to select the suffix to the right of the
5 rightmost "@" symbol. The suffix is a string that is compared with namespace
6 data 306 in trusted domain object records 302 of top level name namespace types
7 304. The selected string is tested for equality, or a substring match, with top level
8 names in trusted domain objects for all trusted forests. If the selected string can be
9 matched with trusted namespaces from two or more trusted forests, the longest
10 substring match is identified as the match. However, the string is not a match if it
11 is equivalent to, or subordinate to, an exclusion record namespace value.

12 For example, three users have usernames userone@office.company.com,
13 usertwo@store.company.com, and userthree@sales.company.com, respectively.
14 The first and second usernames will be equated with the namespace data 306
15 corresponding to FTInfo records one (1) and two (2) in the trusted domain object
16 data structure 300 (Fig. 3). The third username, userthree@sales.company.com,
17 will not be matched to any namespace data 306 in FTInfo records 302 and the
18 authentication request for the third username will fail.

19 A service principal name syntax can be represented as
20 "ServiceType/InstanceName [:PortNum][/ServiceName [@Domain]]." The
21 ServiceType identifies the type of service, such as "www" for a World Wide Web
22 service. The InstanceName can be either a name or an Internet protocol (IP)
23 address of a host computer executing the service. The PortNum identifies the port
24 number of the service. The ServiceName is the name of the service, if different
25 than the InstanceName (as in the case of replicated services where the same

1 service runs on more than one host). The Domain component is the DNS name of
2 the domain that maintains the service account.

3 A service principal name is parsed starting from the right and proceeds until
4 a component match is found, or until the possibilities are exhausted. The Domain,
5 the ServiceName, and the InstanceName are compared with namespace data 306
6 in trusted domain object FTinfo records 302 of top level name namespace types
7 304. The longest substring match is identified as the matching substring. A match
8 with only the Domain component of a service principal name is sufficient to route,
9 or otherwise communicate, an authentication request. If the Domain component
10 of a service principal name is not present, but both the InstanceName and
11 ServiceName exist, then both must match the name corresponding to the
12 authentication request, and both must point to the same trusted forest in the
13 corresponding pointer field 312. Otherwise, the authentication request fails.

14 **Method for Domain Security Identifier Filtering**

15 Fig. 6 illustrates a method to filter domain security identifiers (SIDs) that
16 are received across autonomous network system boundaries. The described
17 method references components of network architecture 200 (Fig. 2) and data
18 structure 300 (Fig. 3). The order in which the method is described is not intended
19 to be construed as a limitation. Furthermore, the method can be implemented in
20 any suitable hardware, software, firmware, or combination thereof.

21 Figure 6 illustrates how SIDs that are returned in authorization data when a
22 user is authenticated can be "filtered". At block 600, a root domain in an
23 independent network system receives a list of SIDs, including the user's account
24 domain SID, plus user and group SIDs, via a trust link from a root domain in a
25 second independent network system. SID-filtering provides a level of security by

1 filtering out SIDs that are not relative to the user's account domain, or that are not
2 relative to any other domain in the user's account forest. SID filtering will prevent
3 the user from being able to access resources based on membership in groups that
4 are instantiated outside of the user's account domain, or outside of the user's
5 account forest. SID filtering can also prevent the user from being able to access
6 resources that grant access based on a previous domain account SID.

7 Additionally, system resource access is protected with access control lists
8 that use security identifiers to identify the security principal (a user or group) that
9 is granted access rights to the resource(s). Access control lists correspond to a
10 resource to indicate which users and groups are permitted to access it, and what
11 level of access they are allowed. When a user requests access to a system
12 resource, the set of SIDs that identify the user and associated group memberships
13 is compared to the access control list.

14 A SID history is an attribute on user and group objects used to hold
15 previous SIDs if that user or group was migrated to the current domain from a
16 different domain. The SID history is a list to track a user or group being migrated
17 multiple times. When a user is authenticated, a domain controller in the user's
18 account domain determines group memberships using both the current user
19 account SID, and any SIDs in the SID history. If the user account has been
20 migrated, access to resources based on the previous account can be maintained.

21 SID filtering is automatically initiated for all of the domains in a user's
22 account forest when SIDs are routed, or otherwise communicated, via a trust link
23 path between the root domains of a trusted forest and a trusting forest in response
24 to an authentication or authorization request for a security principal from the
25 trusted forest. For a Kerberos authentication request, SIDs are filtered when a

1 principal from the trusted forest requests a ticket for a service in the trusting forest.
2 For an NTLM authentication request, SIDs are filtered when a response to the
3 authentication request is returned via a trust link between root domains of the two
4 independent network systems.

5 When a user from a trusted domain logs onto a computer in a trusting
6 domain, the user's current account SID and group membership SIDs are
7 determined by a domain controller in the user's account domain. The group
8 membership SIDs also contain a SID history for the user. Computers in the
9 trusting domain are not able to determine or verify whether the SIDs in the SID
10 history actually correspond to the user account from a time when the user account
11 was located in a different domain. Computers in the trusting domain accept the
12 information from a trusted domain controller as a trusted authority for that user's
13 account information. Membership in domain local groups in the trusting domain
14 will be determined using all the SIDs presented by the user's account domain.
15 Access control decisions in the trusting domain are made accordingly.

16 SIDs are filtered to ensure that trusted forests are allowed to provide only
17 SIDs for which they are trusted to be authoritative. Otherwise, SID information
18 routed from one independent network system to another via a trust link could be
19 altered such that a user from a trusted forest can impersonate any user from a
20 different trusted forest, or from the trusting forest. A security concern arises if a
21 SID is falsely added to a user's SID history. The bogus SID entry may allow that
22 user to gain unauthorized access to resources in a trusting domain, or forest.

23 An operating system, such as Windows®, is designed to rigidly control
24 modification of any SID history attribute to protect against forgery. However,
25 there is no restriction on the value of a SID that could be added to the SID history

1 attribute of any user or group to allow for migration from any previous account
2 domain. When a user is authenticated, all of the SIDs in the user's object,
3 including the SID history attribute, are returned in the authorization data. If
4 conventional operating system protections are disabled or bypassed, a user or
5 group SID, from any domain in any forest, could be forged such that the user will
6 satisfy access control checks and thus be able to gain "unauthorized" access to any
7 protected resource in the trusting domain, or forest.

8 At block 602, the root domain for the independent network system
9 compares a user's account domain SID against a list of trusted domain SIDs
10 compiled from the FTinfo records 302 in the trusted domain object data structure
11 300. If the user's account domain SID is rejected (i.e., "no" from block 604), the
12 authentication request fails at block 606.

13 If the user's account domain SID is accepted (i.e., "yes" from block 604), a
14 received SID is compared against the list of trusted domain SIDs at block 608.
15 Based on the comparison, the root domain determines whether to accept or reject
16 the received SID. Further, only user and group SIDs that are relative to the list of
17 trusted domain SIDs will be accepted, as well as the user's account domain SID
18 itself. This restriction applies to all authentication requests, whether they originate
19 from the trusting domain or are forwarded on behalf of some other domain further
20 along a trust path.

21 If the received domain SID is rejected (i.e., "no" from block 610), the SID
22 is removed from the user object containing the SID at block 612. Domain SIDs
23 are removed if they are not relative to an enabled domain SID component from an
24 FTinfo record for that forest. If the received SID is accepted as trusted (i.e., "yes"
25 from block 610), the SID is accepted as being related to an enabled domain SID

1 component, and therefore trusted at block 614. The process of comparing received
2 user or group SIDs against a list of trusted domain SIDs (block 608), and either
3 determining to accept or not to accept the received SID (blocks 610 through 614),
4 is repeated until all of the SIDs received from a trusted forest are evaluated. After
5 the received SIDs are verified, authorization and access control proceeds just as if
6 the user had been authenticated in the local forest.

7 Alternatively and/or in addition to the method described to filter domain
8 SIDs, a SID history quarantine solution can be implemented that eliminates the
9 risk of a rogue domain administrator in a directly trusted domain utilizing SID
10 history to alter SIDs in a user's authorization data. The quarantine solution
11 deploys a defensive mechanism on a trusting domain which is effective regardless
12 of how an unauthorized SID attack is attempted. All domain controllers in the
13 trusting domain are configured to filter SIDs in any authorization data received
14 from the trusted domain. SID filtering removes any SIDs in the authorization data
15 that are not relative to the trusted domain. The trusted domain that is targeted for
16 SID filtering is considered to be quarantined.

17 A trusting domain can enforce a SID history quarantine against any other
18 domain that it directly trusts. This modifies the processing of authentication
19 requests when users from the quarantined domain request to be logged on. Any
20 domain controller in the trusting domain can determine the correct domain SID for
21 the quarantined domain, and filter the SIDs in the authorization data to remove any
22 that are not relative to that domain. While a given domain can only be quarantined
23 by another domain that directly trusts it, the effect is inherited by any domain
24 further along the trust path in the trusting direction. No changes are required for
25 domain controllers in the trusted domain.

1 The SID history quarantine solution enables a domain controller to protect
2 itself from SID history attacks launched by a rogue administrator in any domain
3 that it directly trusts. The solution allows domain controllers in the trusting
4 domain to be configured to filter SIDs in the authorization data received from any
5 domain controller in the trusted domain. From the trusting domain's perspective,
6 the trusted domain is completely quarantined because the trusted domain can no
7 longer provide SIDs from any other domain, and thus any type of SID history
8 attack will not be successful regardless of how it is implemented.

9 **Exemplary Computing System and Environment**

10 Fig. 7 illustrates an example of a computing environment 700 within which
11 the computer, network, and system architectures described herein can be either
12 fully or partially implemented. Exemplary computing environment 700 is only
13 one example of a computing system and is not intended to suggest any limitation
14 as to the scope of use or functionality of the network architectures. Neither should
15 the computing environment 700 be interpreted as having any dependency or
16 requirement relating to any one or combination of components illustrated in the
17 exemplary computing environment 700.

18 The computer and network architectures can be implemented with
19 numerous other general purpose or special purpose computing system
20 environments or configurations. Examples of well known computing systems,
21 environments, and/or configurations that may be suitable for use include, but are
22 not limited to, personal computers, server computers, thin clients, thick clients,
23 hand-held or laptop devices, multiprocessor systems, microprocessor-based
24 systems, set top boxes, programmable consumer electronics, network PCs,
25

1 minicomputers, mainframe computers, distributed computing environments that
2 include any of the above systems or devices, and the like.

3 Authentication and authorization across autonomous network system
4 boundaries may be described in the general context of computer-executable
5 instructions, such as program modules, being executed by a computer. Generally,
6 program modules include routines, programs, objects, components, data structures,
7 etc. that perform particular tasks or implement particular abstract data types.
8 Authentication and authorization across autonomous network system boundaries
9 may also be practiced in distributed computing environments where tasks are
10 performed by remote processing devices that are linked through a communications
11 network. In a distributed computing environment, program modules may be
12 located in both local and remote computer storage media including memory
13 storage devices.

14 The computing environment 700 includes a general-purpose computing
15 system in the form of a computer 702. The components of computer 702 can
16 include, by are not limited to, one or more processors or processing units 704, a
17 system memory 706, and a system bus 708 that couples various system
18 components including the processor 704 to the system memory 706.

19 The system bus 708 represents one or more of any of several types of bus
20 structures, including a memory bus or memory controller, a peripheral bus, an
21 accelerated graphics port, and a processor or local bus using any of a variety of
22 bus architectures. By way of example, such architectures can include an Industry
23 Standard Architecture (ISA) bus, a Micro Channel Architecture (MCA) bus, an
24 Enhanced ISA (EISA) bus, a Video Electronics Standards Association (VESA)

1 local bus, and a Peripheral Component Interconnects (PCI) bus also known as a
2 Mezzanine bus.

3 Computer system 702 typically includes a variety of computer readable
4 media. Such media can be any available media that is accessible by computer 702
5 and includes both volatile and non-volatile media, removable and non-removable
6 media. The system memory 706 includes computer readable media in the form of
7 volatile memory, such as random access memory (RAM) 710, and/or non-volatile
8 memory, such as read only memory (ROM) 712. A basic input/output system
9 (BIOS) 714, containing the basic routines that help to transfer information
10 between elements within computer 702, such as during start-up, is stored in ROM
11 712. RAM 710 typically contains data and/or program modules that are
12 immediately accessible to and/or presently operated on by the processing unit 704.

13 Computer 702 can also include other removable/non-removable,
14 volatile/non-volatile computer storage media. By way of example, Fig. 7
15 illustrates a hard disk drive 716 for reading from and writing to a non-removable,
16 non-volatile magnetic media (not shown), a magnetic disk drive 718 for reading
17 from and writing to a removable, non-volatile magnetic disk 720 (e.g., a "floppy
18 disk"), and an optical disk drive 722 for reading from and/or writing to a
19 removable, non-volatile optical disk 724 such as a CD-ROM, DVD-ROM, or other
20 optical media. The hard disk drive 716, magnetic disk drive 718, and optical disk
21 drive 722 are each connected to the system bus 708 by one or more data media
22 interfaces 726. Alternatively, the hard disk drive 716, magnetic disk drive 718,
23 and optical disk drive 722 can be connected to the system bus 708 by a SCSI
24 interface (not shown).
25

1 The disk drives and their associated computer-readable media provide non-
2 volatile storage of computer readable instructions, data structures, program
3 modules, and other data for computer 702. Although the example illustrates a
4 hard disk 716, a removable magnetic disk 720, and a removable optical disk 724,
5 it is to be appreciated that other types of computer readable media which can store
6 data that is accessible by a computer, such as magnetic cassettes or other magnetic
7 storage devices, flash memory cards, CD-ROM, digital versatile disks (DVD) or
8 other optical storage, random access memories (RAM), read only memories
9 (ROM), electrically erasable programmable read-only memory (EEPROM), and
10 the like, can also be utilized to implement the exemplary computing system and
11 environment.

12 Any number of program modules can be stored on the hard disk 716,
13 magnetic disk 720, optical disk 724, ROM 712, and/or RAM 710, including by
14 way of example, an operating system 726, one or more application programs 728,
15 other program modules 730, and program data 732. Each of such operating
16 system 726, one or more application programs 728, other program modules 730,
17 and program data 732 (or some combination thereof) may include an embodiment
18 of authentication and authorization across autonomous network system
19 boundaries.

20 Computer system 702 can include a variety of computer readable media
21 identified as communication media. Communication media typically embodies
22 computer readable instructions, data structures, program modules, or other data in
23 a modulated data signal such as a carrier wave or other transport mechanism and
24 includes any information delivery media. The term "modulated data signal"
25 means a signal that has one or more of its characteristics set or changed in such a

1 manner as to encode information in the signal. By way of example, and not
2 limitation, communication media includes wired media such as a wired network or
3 direct-wired connection, and wireless media such as acoustic, RF, infrared, and
4 other wireless media. Combinations of any of the above are also included within
5 the scope of computer readable media.

6 A user can enter commands and information into computer system 702 via
7 input devices such as a keyboard 734 and a pointing device 736 (e.g., a "mouse").
8 Other input devices 738 (not shown specifically) may include a microphone,
9 joystick, game pad, satellite dish, serial port, scanner, and/or the like. These and
10 other input devices are connected to the processing unit 604 via input/output
11 interfaces 740 that are coupled to the system bus 708, but may be connected by
12 other interface and bus structures, such as a parallel port, game port, or a universal
13 serial bus (USB).

14 A monitor 742 or other type of display device can also be connected to the
15 system bus 708 via an interface, such as a video adapter 744. In addition to the
16 monitor 742, other output peripheral devices can include components such as
17 speakers (not shown) and a printer 746 which can be connected to computer 702
18 via the input/output interfaces 740.

19 Computer 702 can operate in a networked environment using logical
20 connections to one or more remote computers, such as a remote computing device
21 748. By way of example, the remote computing device 748 can be a personal
22 computer, portable computer, a server, a router, a network computer, a peer device
23 or other common network node, and the like. The remote computing device 748 is
24 illustrated as a portable computer that can include many or all of the elements and
25 features described herein relative to computer system 702.

1 Logical connections between computer 702 and the remote computer 748
2 are depicted as a local area network (LAN) 750 and a general wide area network
3 (WAN) 752. Such networking environments are commonplace in offices,
4 enterprise-wide computer networks, intranets, and the Internet. When
5 implemented in a LAN networking environment, the computer 702 is connected to
6 a local network 750 via a network interface or adapter 754. When implemented in
7 a WAN networking environment, the computer 702 typically includes a modem
8 756 or other means for establishing communications over the wide network 752.
9 The modem 756, which can be internal or external to computer 702, can be
10 connected to the system bus 708 via the input/output interfaces 740 or other
11 appropriate mechanisms. It is to be appreciated that the illustrated network
12 connections are exemplary and that other means of establishing communication
13 link(s) between the computers 702 and 748 can be employed.

14 In a networked environment, such as that illustrated with computing
15 environment 700, program modules depicted relative to the computer 702, or
16 portions thereof, may be stored in a remote memory storage device. By way of
17 example, remote application programs 758 reside on a memory device of remote
18 computer 748. For purposes of illustration, application programs and other
19 executable program components, such as the operating system, are illustrated
20 herein as discrete blocks, although it is recognized that such programs and
21 components reside at various times in different storage components of the
22 computer system 702, and are executed by the data processor(s) of the computer.

23 Conclusion

24 The systems and methods described herein facilitate an enterprise having
25 independent business units for administrative autonomy or asset isolation, yet

1 allow users and administrators in one forest to obtain frequently needed
2 authenticated and authorized access to servers in another forest. Kerberos
3 authentication requests that succeed when the user and service have accounts
4 managed in different domains, but the same network system (e.g., a single forest),
5 will also succeed across an autonomous security boundary when the accounts are
6 managed in different network systems (e.g., independent forests).

7 Although the systems and methods have been described in language
8 specific to structural features and/or methodological steps, it is to be understood
9 that the technology defined in the appended claims is not necessarily limited to the
10 specific features or steps described. Rather, the specific features and steps are
11 disclosed as preferred forms of implementing the claimed invention.
12
13
14
15
16
17
18
19
20
21
22
23
24
25